Manual de SSC-32.

Controlador de servos de 32 canales SCC32 S310185

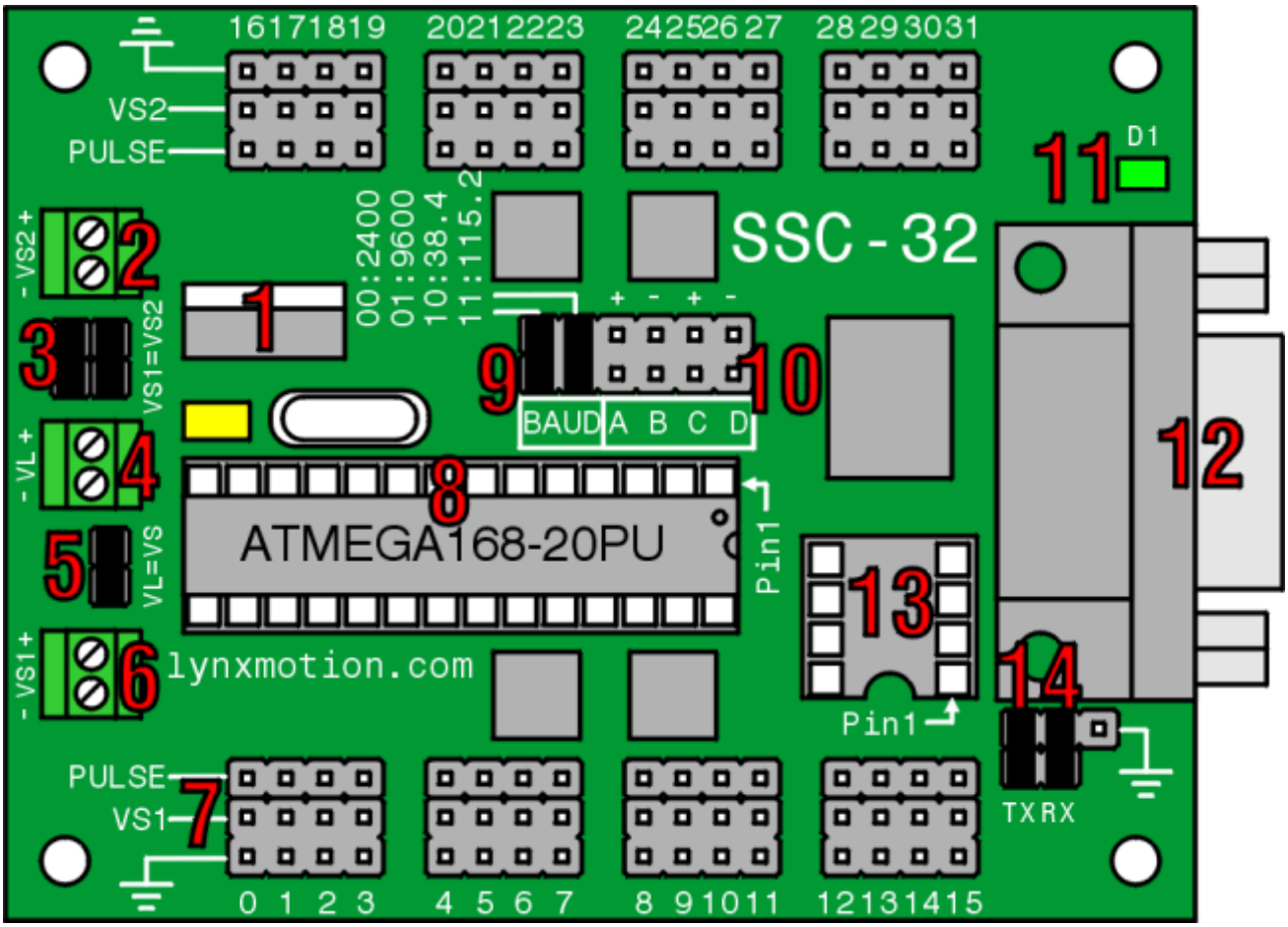
Autor: Jim Frye
Versión : 2.01XE
Fecha: febrero 18, 2009
Traducción al español: Alicia Bernal
www.Superrobotica.com

Índice

Enlaces

- Esquema de SSC-32 (pdf)
- Descarga de LynxTerm

SSC-32.



Información de hardware de SSC-32.

1. El regulador de tensión de baja caída (LDO) proporcionará una salida de 5 voltios CC con una entrada de tan sólo 5,5 voltios CC. Este dato es importante al alimentar su robot a través de una batería. Admite una entrada máxima de 9 voltios CC. El regulador tiene una potencia nominal de 500 mA, aunque debe intentar reducirlo a 250 mA para evitar que el regulador se sobrecaliente en exceso.
2. Este terminal es la alimentación de los canales de los servos (del 16 al 31). Se deben aplicar de 4,8 a 6 voltios CC para los servos analógicos y digitales. Se pueden obtener directamente de un pack de pilas de 5 pilas NiMH. Los servos HSR-5980 o HSR-5990 se pueden alimentar con 7,2 vdc - 7,4 vdc. Se pueden obtener directamente de un pack de 2 pilas de litio o un pack de 2 pilas de NiMH.

Placa	Entrada
VS2+	ROJO
VS1 -	NEGRO

3. Estos jumpers se utilizan para conectar VS1 a VS2. Utilice esta opción cuando vaya a alimentar todos los servos desde la misma batería. Use ambos jumpers. Si desea utilizar dos packs de pilas independientes, uno en cada lado; elimine estos dos jumpers.
4. Esta es la tensión lógica o VL Es la alimentación para la electronica del circuito. Esta entrada se utiliza normalmente con un conector de batería de 9 voltios CC para alimentar los ICs y cualquier cosa que esté conectada a las líneas de 5 voltios CC de la placa. El rango válido para este terminal es 6 vdc - 9 vdc. Esta entrada se utiliza para aislar la alimentación de la lógica de la alimentación de los servos. Es necesario eliminar el jumper VS1=VL si los servos se van a alimentar de forma independiente desde los conectores VS . El circuito SSC-32 debería consumir 35 mA sin tener nada conectado a la salida de 5 vdc.

Placa	Entrada
VL+	ROJO
VL -	NEGRO

5. Este jumper permite alimentar el microcontrolador y los servos desde la misma alimentación del conector VL. Se

requieren al menos 6 voltios CC para que funcione correctamente. Si el microcontrolador se resetea cuando haya varios servos en movimiento, entonces es recomendable alimentar el microcontrolador de forma independiente a través de la entrada VL. Una batería de 9 voltios CC es perfectamente adecuada para ello. Este jumper debería eliminarse si se va a alimentar el microcontrolador de forma independiente.

6. Este terminal es la alimentación de los canales de los servos (del 0 al 15). Se deben aplicar de 4,8 a 6 voltios CC para los servos analógicos y digitales. Se pueden obtener directamente de un pack de pilas de 5 pilas NiMH. Los servos HSR-5980 o HSR-5990 se pueden alimentar con 7,2 vdc - 7,4 vdc. Se pueden obtener directamente de un pack de 2 pilas de litio o un pack de 2 pilas de NiMH.

Placa	Entrada
VL1+	ROJO
VL1 -	NEGRO

7. Aquí es donde tiene que conectar los servos y los demás dispositivos de salida. Con cuidado, desactive la alimentación al conectar cualquier elemento al bus de E/S.

Placa	Cable
Pulso	Amarillo o blanco
VS	Rojo
Tierra	Negro o marrón

8. Aquí es donde se encuentra el chip IC de Atmel. Deberá insertar con cuidado el Pin 1 con la esquina superior derecha como se indica en la imagen. Evite doblar los pines.

9. Las dos entradas de BAUDIOS (BAUD) permiten configurar la tasa de baudios. Vea los ejemplos siguientes.

Jumpers	Tasa de baudios	Uso
0 0	2400	Procesadores de menor velocidad
0 1	9600	Procesadores de menor velocidad
1 0	38,4k	Comunicación Atom/Stamp
1 1	115,2k	Comunicación con el PC, Actualización de firmware

10. Las entradas ABCD tienen soporte estático y biestable. Las entradas tienen resistencias internas de tipo pullup (50k) débiles que se utilizan con los comandos de entrada digital de lectura. Le recomendamos que utilice un interruptor normalmente abierto desde la entrada a tierra.

11. Este es el LED indicador del buen estado del procesador. Se iluminará de forma continua cuando se aplique la alimentación y permanecerá iluminado hasta que el procesador haya recibido un comando serie válido. Se apagará y volverá a parpadear siempre que reciba datos serie.

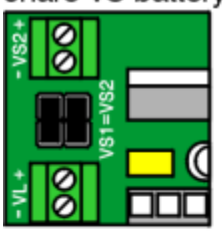
12. Simplemente debe conectar un cable con conector DB9 Macho/Hembra desde este conector a un puerto serie libre de 9 pines de su ordenador para recibir los datos de posicionamiento de los servos. También puede utilizar un adaptador de USB a puerto serie. Tenga en cuenta que existen numerosos adaptadores USB-SERIE que requieren una alimentación independiente para funcionar correctamente.

13. Se trata de un zocalo para una EEPROM de 8 pines. La EEPROM es compatible con el firmware 2.01GP.

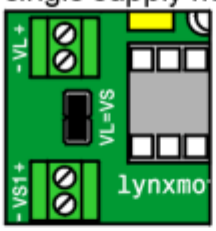
14. Este es el puerto serie a nivel TTL. Instale dos jumpers como se muestra a continuación para habilitar el puerto DB9. Instale dos conectores para utilizar la comunicación serie a nivel TTL desde un microcontrolador.

Conexión de jumpers y conectores

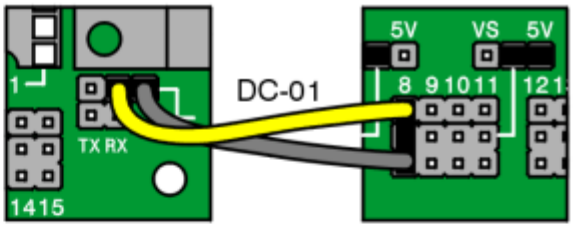
Applies VS1 to VS2 share VS battery.



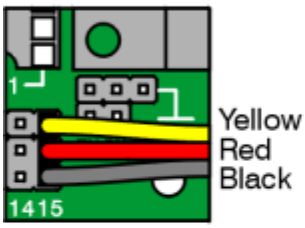
Applies VS to VL single supply mode.



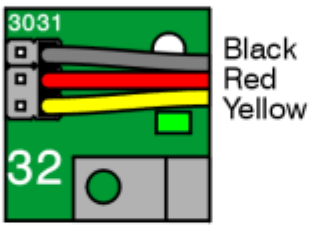
Unidirectional TTL Serial Communications.
SSC-32 side...
Bot Board side...



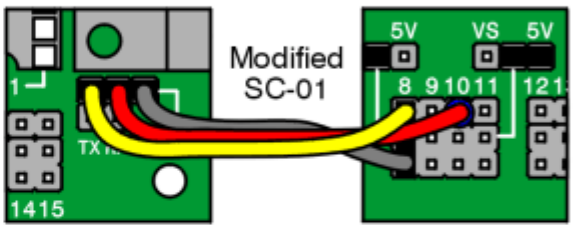
Example servo connection 0-15.



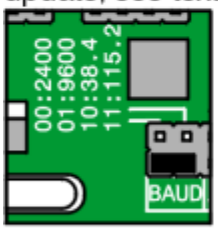
Example servo connection 16-31.



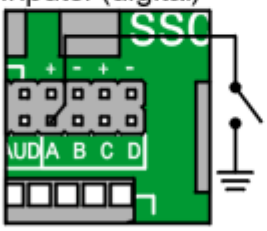
Bidirectional TTL Serial Communication.
SSC-32 side...
Bot Board side...



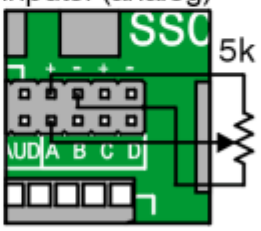
Force firmware update, see text!



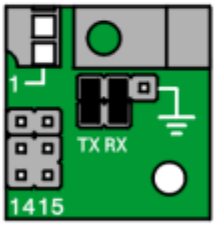
ABCD auxiliary inputs. (digital)



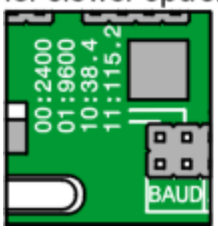
ABCD auxiliary inputs. (analog)



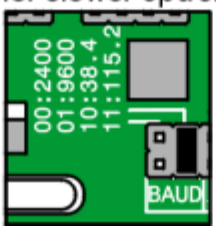
DB9 enable for PC use.



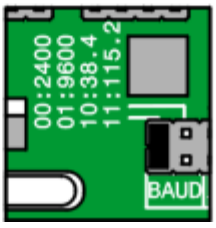
Baud rate 2400 for slower cpu's.



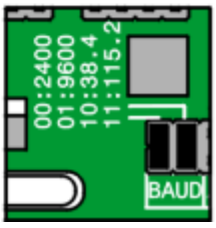
Baud rate 9600 for slower cpu's.



Baud rate 38.4k for Basic Atom use.



Baud rate 115.2k for PC use.



Formato de comandos para SSC-32

Tipos y grupos de comandos

Salvo el modo MiniSSC-II, todos los comandos SSC-32 deben terminar en un carácter de retorno de carro (ASCII 13). Se pueden ejecutar varios comandos del mismo tipo de forma simultánea en un *Grupo de comandos*. Todos los comandos incluidos en un grupo de comandos se ejecutarán después de recibir el retorno de carro. Los comandos de tipos diferentes no pueden combinarse dentro del mismo grupo de comandos. Además, los argumentos numéricos para todos los comandos SSC-32 deben ser cadenas ASCII de los números decimales, por ejemplo, "1234". Algunos comandos admiten números negativos, por ejemplo, "-5678". Se proporcionarán ejemplos de programación más adelante. El formato de ASCII no es sensible al uso de mayúsculas/minúsculas. Use cuantos bytes sean necesarios. Se ignorarán los espacios, tabuladores y saltos de líneas.

Tipos y grupos de comandos			
1	Movimiento de servos	7	Lectura de entradas analógicas
2	Salida discreta	8	Secuenciador de forma de caminar del hexapod de 12 servos
3	Salida de bytes	9	Consultar secuenciador de hexapod
4	Consulta de estado de movimiento	10	Obtener versión
5	Consulta de ancho de pulso	11	Ir a arranque
6	Lectura de salidas digitales	12	Compatibilidad de MiniSSC-II

Mover servo o mover grupo

# <ch> P <pw> S <spd> ... # <ch> P <pw> S <spd> T <time> <cr>	
<ch>	Número de canal en formato decimal, 0-31
<pw>	Ancho de pulso en microsegundos, 500-2500
<spd>	Velocidad de movimiento en uS por segundos para cada canal (opcional)
<time>	Tiempo en mS para el movimiento completo, que afecta a todos a los canales, 65535 máx (opcional)
<cr>	Carácter de retorno de carro, ASCII 13 (necesario para iniciar la acción)
<esc>	Cancela la acción actual, ASCII 27

Ejemplo de movimiento de servo: "#5 P1600 S750 <cr>"

El ejemplo moverá el servo en el canal 5 a la posición 1600. Se moverá desde su posición actual a una velocidad de 750uS por segundo hasta que alcance el destino del comando. Para comprender mejor el argumento de velocidad, tenga en cuenta que un desplazamiento de 1000 uS equivale a un giro de 90° aproximadamente. Un valor de velocidad de 100 uS por segundo significa que el servo tardará 10 segundos en girar 90°. Asimismo, un valor de velocidad de 2000 uS por segundo equivale a 500 mS (medio segundo) para girar 90°.

Ejemplo de movimiento de servo: "#5 P1600 T1000 <cr>"

El ejemplo moverá el servo 5 a la posición 1600. Tardará 1 en completar el movimiento independientemente de lo lejos que deba desplazarse el servo para alcanzar el destino.

Ejemplo de movimiento de grupo de servos: "#5 P1600 #10 P750 T2500 <cr>"

El ejemplo moverá el servo 5 a la posición 1600 y el servo 10 a la posición 750. Tardará unos 2,5 segundos en completar el movimiento, aunque uno de los servos tenga que desplazarse más lejos que el otro. Ambos servos empezarán y dejarán de moverse al mismo tiempo. Este es un comando muy potente. Mediante este comando todas las patas de un robot caminante se sincronizan a la perfección con el movimiento de grupo. El mismo movimiento sincronizado sirve para controlar un brazo robótico también.

Puede combinar los comandos de velocidad y tiempo como desee. La velocidad para cada servo se calculará dependiendo de las siguientes reglas:

1. Todos los canales empezarán y dejarán de moverse de forma simultánea.
2. Si se especifica la velocidad para un servo, éste no se moverá más rápido que la velocidad especificada (pero podría moverse más lento si el comando de tiempo así lo requiere).

3. Si se especifica un tiempo para el movimiento, entonces el movimiento tardará como mínimo el tiempo especificado (pero puede tardar más si la velocidad del comando así lo requiere).

Ejemplo de movimiento de servo: "#5 P1600 #17 P750 #2 P2250 T2000 <cr>"

El ejemplo proporciona 1600 uS en el canal 5, 750 uS en el canal 17 y 2250 uS en el canal 2. El movimiento completo tardará al menos 2 segundos, pero el canal 17 no se moverá más rápido de 500 uS por segundo. El tiempo real del movimiento dependerá del ancho de pulso inicial para el canal 17. Suponga que el canal 17 comienza en la posición 2000. Después se tiene que mover 1250 uS. Dado que está limitado a 500 uS por segundo, necesitará al menos 2,5 segundos, por lo que el movimiento completo tardará 2,5 segundos. Por otro lado, si el canal 17 comienza en la posición 1000, sólo necesitará 250 uS, que se puede hacer en 0,5 segundos, por lo que el movimiento completo tardará 2 segundos.

Importante El primer comando de posicionamiento debería ser un comando "# <ch> P <pw>" normal. Dado que el controlador no sabe dónde se encuentra el servo en el encendido, ignorará los comandos de velocidad y tiempo hasta que se reciba el primer comando normal.

Cualquier movimiento que incluya más de un servo y utilice un modificador S o T se considerará como un movimiento de grupo y todos los servos empezarán y dejaran de moverse al mismo tiempo. Si necesita mover varios servos con velocidades diferentes, debe emitir los comandos por separado.

Software de compensación de posición

# <ch> PO <offset value> ... # <ch> PO <offset value> <cr>	
<ch>	Número de canal en formato decimal, 0-31
<offset value>	100 a -100 en uSegundos
<cr>	Carácter de retorno de carro, ASCII 13

Este comando permite alinear perfectamente la posición centrada (1500 uS) de todos los servos. El canal del servo se compensará en la cantidad expresada por el valor de compensación. Aproximadamente equivale a 15° de rango. Es importante tratar de centrar lo máximo posible los servos antes de aplicar la compensación de los servos. De esta forma resulta más fácil configurar los servos que no tienen una alineación mecánica. El comando de compensación de la posición debería emitirse una sola vez en el programa. Después de desactivar el SSC-32, se anularán las compensaciones de la posición.

La versión actual de SSC-32 dispone ahora de un método de registro interno para las compensaciones de posición. Se almacenan en la memoria EEPROM interna de los chips Atmel y se retienen aunque se desactive la alimentación. En la sección Soporte de registro de este manual podrá encontrar más información de esta característica.

Información general de salidas

Las salidas de SSC-32 proceden de los 4 chips de registro de desplazamiento de 8 bits 74HC595. Hay cuatro bancos de salidas de 8 bits que se muestran como 0-7, 8-15, 16-23 y 24-32. Las salidas pueden descender o alcanzar los 25 mA por pin, aunque se debe reservar un máximo de 70mA por banco.

Salida discreta

# <ch> <lvl> ... # <ch> <lvl> <cr>	
<ch>	Número de canal en formato decimal, 0-31
<lvl>	El nivel lógico para el canal es 'H' para (Alto) y 'L' para Bajo
<cr>	Carácter de retorno de carro, ASCII 13

El canal cambiará al nivel indicado dentro de un tiempo de 20 mS tras la recepción del retorno de carro.

Ejemplo de salida discreta: "#3H #4L <cr>"

Este ejemplo muestra una salida ALTA (+5v) en el canal 3 y BAJA (0v) en el canal 4.

Salida de bytes

# <bank> : <value> <cr>	
<bank>	(0 = Pines 0-7, 1 = Pines 8-15, 2 = Pines 16-23, 3 = Pines 24-31)
<value>	El valor decimal para la salida del banco seleccionado (0-255), Bit 0 = LSB del banco
<cr>	Carácter de retorno de carro, ASCII 13

Este comando permite escribir 8 bits de datos binarios de una vez. Todos los pines se actualizan de forma simultánea. Los bancos se actualizan a los 20mS de recibir el retorno de carro.

Ejemplo de salida de banco: "#3:123 <cr>"

Este ejemplo generará una salida con un valor 123 (decimal) para el banco 3. 123 (dec) = 01111011 (bin), y el banco 3 se corresponde con los pines 24-31. De esta forma este comando emitirá una salida de "0" para los pines 26 y 31, y una salida de "1" para el resto de los pines.

Consulta de estado de movimiento

Ejemplo: "Q <cr>"

Retornará un "." si el movimiento anterior está completo o un "+" o si está en progreso.

Habrá un retardo de 50uS a 5mS antes de enviar la respuesta.

Consulta de ancho de pulso

Ejemplo: "QP <arg> <cr>"

Devolverá un único byte (en formato binario) que indica el ancho del pulso del servo seleccionado con una resolución de 10uS. Por ejemplo, si el ancho de pulso es 1500 uS, el byte retornado será 150 (binario).

A través del mismo comando se puede obtener información de varios servos. El valor devuelto será un byte por servo. Habrá un retardo entre 50 uS y 5 mS antes de enviar la respuesta. Generalmente, la respuesta se inicia en un periodo de 100 uS.

Lectura de salidas digitales

Ejemplo: "A B C D AL BL CL DL <cr> "

A, B, C y D son las lecturas de entradas normales. Leen el valor de la entrada como un valor binario. Devuelve el valor ASCII "0" si la entrada es de nivel lógico bajo (0 v) o un valor ASCII "1" si la entrada es de nivel lógico alto (+5 v).

AL, BL, CL y DL con las lecturas de entradas multiplexadas. Devuelven el valor en la entrada como un ASCII "0" si la entrada es de valor lógico bajo (0 v) o si ha sido bajo desde el último comando *L. Devuelve un valor lógico alto (+5v) si la entrada es alta y nunca ha bajado desde el último comando *L. Es decir, devolverá un valor lógico bajo si la entrada nunca baja. La lectura del estado simplemente resetea la entrada multiplexada.

Las entradas ABCD tienen una resistencia pullup (~50k) débil que se activa cuando se utiliza como entrada. Se comprueban aproximadamente cada 1 mS y rebotan cada 15 mS aproximadamente. El valor lógico para los comandos de lectura no cambiará hasta que la entrada haya permanecido durante 15 mS en el nuevo nivel lógico. Los comandos de lectura de entrada digital pueden agruparse en una sola lectura, con hasta 8 valores por lectura. Devolverá una cadena con un carácter por entrada sin espacios.

Ejemplo de lectura de entrada digital: "A B C DL <cr>"

Lectura de entradas analógicas

Ejemplo: "VA VB VC VD <cr>"

VA, VB, VC y VD leen el valor de la entrada como analógico. Devuelve un solo byte con el valor de 8 bits (binario) para la tensión del pin.

Cuando se utilizan las entradas ABCD como entradas analógicas, se desactiva la resistencia pullup interna. Las entradas se filtran digitalmente para reducir el efecto del ruido. Los valores filtrados adoptarán sus valores finales a los 8 mS del cambio. Un valor de retorno de 0 representa 0 v cc. Un valor de retorno de 255 representa +4,98 v cc. Para convertir el valor devuelto a una tensión, multiplique por 5/256. En el encendido, las entradas ABCD están configuradas para la entrada digital con resistencia pullup. La primera vez que se use un comando V*, el pin se convertirá a una entrada analógica sin resistencia pullup. El resultado de esta primera lectura no devolverá un dato válido.

Ejemplo de lectura de entrada analógica: "VA VB <cr>"

Este ejemplo devolverá 2 bytes con los valores analógicos de A y B. Por ejemplo, si la tensión en el Pin A es 2 v cc y en el Pin B es 3,5 v cc, el valor devuelto será los bytes 102 (binario) y 179 (binario).

Comandos del secuenciador del hexapod para 12 servos

LH <arg>, LM <arg>, LL <arg>

Configure el valor para los servos verticales de la izquierda del hexapod. LH define el valor alto, es decir, el ancho de pulso para levantar la pata a su máxima altura; LM define el valor medio; y LL define el valor bajo. El rango válido para los argumentos es 500 -2500 uS.

RH <arg>, RM <arg>, RL <arg>

Configure el valor para los servos verticales de la derecha del hexapod. RH define el valor alto, es decir, el ancho de pulso para levantar la pata a su máxima altura; RM define el valor medio; y RL define el valor bajo. El rango válido para los argumentos es 500 -2500 uS.

VS <arg>

Define la velocidad de movimiento de los servos verticales. Todos los movimientos de los servos verticales utilizan esta velocidad. El rango válido es 0 - 65535 uS/seg.

LF <arg>, LR <arg>

Configure el valor para los servos horizontales de la izquierda del robot. LF define el valor frontal, es decir, el ancho de pulso para mover la pata a la posición delantera máxima; LR define el valor posterior. El rango válido para los argumentos es 500 - 2500 uS.

RF <arg>, RR <arg>

Configure los valores para los servos horizontales de la derecha del robot. RF define el valor frontal, es decir, el ancho de pulso para mover la pata a la posición delantera máxima; RR define el valor posterior. El rango válido para los argumentos es 500 - 2500 uS.

HT <ara>

Define el tiempo de desplazamiento entre las posiciones horizontales frontal y posterior. El rango válido para los argumentos es 1 - 65535 uS.

XL <arg>, XR <arg>

Define el porcentaje de desplazamiento para las patas izquierdas y derechas. El rango válido es desde -100% hasta 100%. Los valores negativos hacen que las patas se muevan hacia atrás. Con un valor de 100%, las patas se moverán entre las posiciones delanteras y traseras. Valores inferiores al 100% permiten un desplazamiento proporcionalmente inferior, pero siempre centrado. La velocidad para los desplazamientos horizontales se ajustan basándose en los comandos XL y XR, por lo que el tiempo de desplazamiento sigue siendo el mismo.

XS <arg>

Ajuste el porcentaje de la velocidad horizontal para todas las patas. El rango válido es desde 0% hasta 200%. Con un valor del 100%, el tiempo de desplazamiento horizontal se corresponderá con el valor programado en el comando HT. Los valores superiores reducen proporcionalmente el tiempo de desplazamiento, mientras que los valores inferiores lo aumentan. Un valor de 0% detendrá el robot en la posición. El secuenciador del hexapod no se iniciará hasta que no se reciba el comando XS.

XSTOP

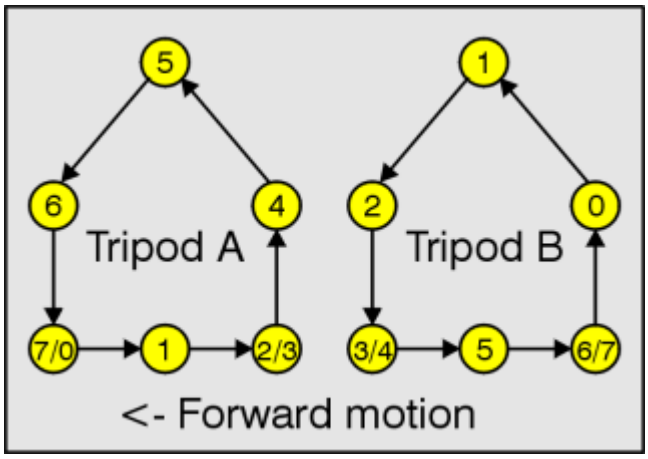
Pare el secuenciador de hexapod. Retorna todos los servos a su funcionamiento normal.

Notas sobre el secuenciador del hexapod

1. Los siguientes canales de servos se utilizan para el secuenciador hexapod:

0	Vertical posterior derecha	16	Vertical posterior izquierda
1	Horizontal posterior derecha	17	Horizontal posterior izquierda
2	Vertical central derecha	18	Vertical central izquierda
3	Horizontal central derecha	19	Horizontal central izquierda
4	Vertical frontal derecha	20	Vertical frontal izquierda
5	Horizontal frontal derecha	21	Horizontal frontal izquierda

2. El hexapod camina alternando tres patas. Los pasos a tres patas se denominan Trípode A y Trípode B. Trípode A incluye {Frontal izquierda, Posterior izquierda y Central derecha}, y Trípode B incluye {Central izquierda, Frontal derecha, Posterior derecha}.
3. Al caminar, las patas pasan por 6 puntos: (Frontal inferior), (Central inferior), (Posterior inferior), (Posterior medio), (Central superior), y (Frontal medio). "Central" se refiere a un punto medio entre las posiciones frontal y posterior.



4. La secuencia del caminar incluye 8 estados, numerados 0-7. A continuación se definen estos estados:

Estado	Trípode A		Trípode B	
	Vertical	Horizontal	Vertical	Horizontal
0	Baja	Frontal a Central	Media a Alta	Posterior a Central
1	Baja	Central a Posterior	Alta a Media	Central a Frontal
2	Baja	Parte trasera	Media a Baja	Parte delantera
3	Baja a Media	Parte trasera	Baja	Parte delantera
4	Media a Alta	Posterior a Central	Baja	Frontal a Central
5	Alta a Media	Central a Frontal	Baja	Central a Posterior
6	Media a Baja	Parte delantera	Baja	Parte trasera
7	Baja	Parte delantera	Baja a Media	Parte trasera

   En esta tabla, "Frontal" y "Posterior" se modifican por los comandos XL y XR. Un valor del 100% tiene como resultado el movimiento especificado en la tabla. Entre 0 y 100%, las posiciones frontal/posterior se acercan a la posición central. Para los valores negativos, la posición Frontal y Posterior se intercambian. Por ejemplo, con un comando XL de -100%, en Estado 0, el Trípode A del lado izquierdo se movería desde la posición Posterior a la Central, y el Trípode B se movería desde la posición Frontal a la Central.
5. Cuando un servo horizontal se está moviendo, su velocidad se ajustará basándose en los anchos de pulso de Frontal/Posterior, el porcentaje de XL/XR, y el porcentaje de XS. Independientemente de la distancia del desplazamiento entre la posición frontal a la posterior (ajustado por XL/XR), el tiempo de desplazamiento total será HT dividido entre el porcentaje de XS.
6. Cuando un servo vertical se esté desplazando desde la posición Inferior a Media o desde Media a Baja, se moverá a la velocidad especificada por el comando VS. Cuando un servo vertical se esté desplazando desde la posición Media a Baja o Alta a Media, la velocidad vertical se ajustará para que los movimientos horizontales y verticales finalicen al

mismo tiempo.

7. Se puede generar cualquiera de los comandos del secuenciador del hexapod mientras que el secuenciador esté en funcionamiento. Tendrá efecto de forma inmediata.

Ejemplos del secuenciador del hexapod:

"LH1000 LM1400 LL1800 RH2000 RM1600 RL1200 VS3000 <cr>"

Define los parámetros del servo vertical.

"LF1700 LR1300 RF1300 RR1700 HT1500 <cr>"

Define los parámetros del servo horizontal.

"XL50 XR100 XS100 <cr>"

Hace que gire hacia la izquierda gradualmente al 100% de la velocidad (e inicia el secuenciador si aún no se ha iniciado).

"XL -100 XR 100 XS 50 <cr>"

Hace que gire hacia la izquierda sobre su sitio al 50% de la velocidad.

"XSTOP <CR>"

Detiene el secuenciador y permite controlar los canales 0-5, 16-21 de los servos a través de comandos de servo normales.

Consulta del estado del secuenciador del hexapod

XQ <cr>

Devuelve un 1 dígito que representa el estado del secuenciador del hexapod y el porcentaje aproximado del movimiento en el estado. El valor alto será de '0' a '7', y el valor bajo será de '0' a '9'. Por ejemplo, si el secuenciador está al 80% de desplazamiento hasta el estado 5, devolverá el valor 58 (hex).

Obtener versión de software

VER <cr>

Devuelve el número de versión del software como una cadena ASCII.

Actualización de firmware

La mejor forma de actualizar el firmware es a través de Lynxterm o uno de nuestros paquetes de software. Las instrucciones detalladas sobre el método de actualización del firmware se incluyen en los manuales de software. La actualización de firmware no funcionará si se utiliza la posición del jumper "Forzar actualización de firmware" (como se indica en la sección Conexión de jumpers y conectores) cuando está activada la actualización de firmware normal. No haga nada si no está seguro de saber lo que está haciendo.

Transferir a arranque

GOBOOT <cr>

Inicia el cargador de arranque para ejecutar las actualizaciones del software. Para salir del cargador de arranque y comenzar a utilizar la aplicación, el ciclo de alimentación o introducir (sensible al uso de mayúsculas/minúsculas, sin espacios):

"g0000<cr>"

Emulación de Mini SSC-II

Formato binario, 3-bytes. Recuerde que el circuito SSC-II original funciona a una velocidad de 9600 baudios. Si va a utilizar el SCC-32 con programas escrito para el SCC-II, es muy posible que tenga que cambiar la velocidad del SCC-32 a 9600 baudios para que funcione correctamente.

Byte 1	255, byte de sincronización
Byte 2	0 - 31, el número de servo
Byte 3	0 - 250, el ancho de pulso; 0=500uS, 125=1500uS, 250=2500uS

Registro de SSC-32

Información general de los registros

Número	Nombre	Mínimo	Máximo	Valores por defecto	Descripción
0	Habilitar	0	65535	0	Un campo de bits (16 bits) que habilita varias funciones del SSC-32.
1	Retardo de transmisión	0	65535	600	El retardo, en microsegundos, antes de transmitir el primer byte de una respuesta desde SSC-32.
2	Velocidad de transmisión	0	65535	70	El retardo, en microsegundos, entre los bytes en una respuesta desde el SSC-32.
3-31	(Reservado)	--	--	--	--
32-63	Compensación de pulso inicial	-100	100	0	El valor inicial de la compensación del pulso (PO) para cada servo. El registro 32 se corresponde con el servo #0, el registro 33 con el servo #1, etc.
64-95	Ancho de pulso inicial	0	65535	1500	El valor inicial del ancho del pulso para cada servo. El registro 64 se corresponde con el servo #0, el registro 65 con el servo #1, etc. Un valor de 0 deja la salida del servo con un valor lógico continuo '0'; un valor de 65535 deja una salida de servo con un valor lógico continuo de '1'. Los demás valores se reducen al intervalo de 500 - 2500 microsegundos.
96-255	(Reservado)	--	--	--	--

Nota: Los registros 0-15 son para un uso general, que afectan a todas las operaciones de SSC-32; los registros 32-255 son para la configuración de los canales de los servos individuales, y por ello hay grupos de 32 registros.

Permitir definiciones de bits de registro (R0)

Bit	Nombre	Definición
12 (msb)	Desactivación general	Si '1', se desactivan todas las funciones controladas por el registro Activar. Si '0', se utilizarán los valores de bit individuales para activar o desactivar las funciones.
14-4	(Reservado)	--
3	Activar ancho de pulso inicial	Si '1', se activan los valores de registro de Ancho de pulso inicial en el inicio. Si '0', se utilizará el valor por defecto de 0.
2	Activación de compensación de pulso inicial	Si '1', se activan los valores de registro de compensación de pulso inicial en el inicio. Si '0', se utilizará el valor por defecto de 0.
1	Activar Retardo transmisión/Velocidad	Si '1', se activan los valores de Retardo y Velocidad de transmisión. Si '0', se utilizarán los valores por defecto de 600uS y 70uS respectivamente.
0 (lsb)	Activar cadena de inicio	Si '1', se activa la ejecución de la cadena de inicio cuando se aplica la alimentación a SSC-32. Si '0', la cadena de inicio no se ejecutará.

Lectura/Escritura de registros

Comando	Argumento	Descripción	Ejemplos
Escritura de registro: R <r> = <n> <cr>	r = número de registro, 0-255 n = valor	Programa el valor de un registro. Los espacios son opcionales alrededor del número y valor del registro.	R0=1023 <cr> R32 = -50 <cr>
Lectura de registro: R <r> <cr>	r = número de registro, 0-255	Muestra el valor de un registro, seguido de un retorno de carro. El valor mostrado es en formato ASCII, y se termina con un retorno de carro.	R0 <cr> resultado: 1023<cr> R32 <cr> resultado: -50<cr>
Establece los valores por defecto: RDFLT <cr>	ninguno	Establece todos los registros en los valores por defecto. Cuando el comando ha finalizado, el SSC-32 transmitirá la cadena OK<cr>.	RDFLT <cr> resultado: OK<cr>

El comando RDFLT puede tardar un segundo en ejecutar. No se debería invocar mientras está activo uno de los movimientos programados o la secuencia. No se realizan escrituras de registro hasta que finalice RDFLT (indicado por la respuesta 'ok').

Si el software envía varios comandos R, le recomendamos que el software lea el valor de cada registro después de la escritura. De esta forma se asegurará que finalice la escritura de cada registro después de que se inicie la siguiente.

Si un comando RDFLT o R= está en ejecución, no debe apagar el SSC-32 hasta que haya finalizado. Para determinar si el comando ha finalizado, lea un valor de registro.

Cada vez que se escribe un registro, la ubicación de la memoria EEPROM utilizada para almacenar el valor, experimentará un pequeño gasto de espacio. El número máximo de escrituras es 100.000. No programe su software para que cambie rápidamente los valores de registro, ya que se podría dañar la memoria EEPROM en el procesador ATMega168.

Varios comandos de registro

Comando	Argumento	Descripción	Ejemplos
STOP <n> <cr>	0-31	Detiene inmediatamente el servo especificado en su posición actual. Un espacio es opcional antes del número de servo.	STOP0 <cr> STOP 31 <cr>

Si el servo está realizando un movimiento programado, los demás servos continuarán moviéndose y un comando de consulta indicará que el movimiento continúa hasta que haya transcurrido el tiempo total del movimiento original. Esto es así aunque *todos* los servos del movimiento original se detengan.

Los comandos EER y EEW ya no acceder a la memoria EEPROM interna. Se sustituyen por los comandos de escritura/lectura de registro y cadena de inicio. EER y EEW siguen funcionando para la memoria EEPROM externa.

Cadenas de inicio

Comando	Argumento	Descripción / Ejemplos
Eliminar caracteres: SSDEL <n> <cr>	0-255	Elimina <n> caracteres del final de la cadena de inicio. Si <n> es mayor que la longitud de la cadena de inicio, entonces SSDEL elimina toda la cadena. SSDEL 5 <cr> - Elimina los últimos 5 caracteres de la cadena de inicio SSDEL 255 <cr> - Elimina toda la cadena completa
Concatenar: SSCAT <string> <cr>	Hasta 100 caracteres ASCII	Concatena <string> a la cadena de inicio actual. El espacio en blanco inmediatamente después de "SSCAT" se ignora, pero el resto de los espacios forman parte de la cadena. La cadena se termina con un retorno de carro y no puede contener retornos de carro internos. Los comandos de la cadena de inicio se terminan con un punto y coma (incluyendo el último comando). SSCAT #0P1000#1P2000T3000;<cr> SSCAT PL0 SQ5 SM50;<cr>
Mostrar la cadena de inicio: SS <cr>	ninguno	Muestra la cadena de inicio completa, rodeada por comillas y seguida de un retorno de carro. SS <cr> result: "#0P1000#1P2000T3000;PL0 SQ5 SM50;"<cr>

La cadena de inicio programada se ejecuta durante el arranque de SSC-32, si el bit de activación de Cadena de inicio está establecido en el registro Activar. La cadena de inicio se ejecuta después de aplicar cualquiera de los valores de registro (por ejemplo, ancho de pulso inicial).

La longitud máxima de la cadena de inicio es 100 caracteres ASCII. Se ignorarán todos los caracteres adicionales.

Los siguientes comandos no deberían utilizarse en una cadena de inicio: EER, EEW, R=, SSCAT, SSDEL.

El comando SS puede tardar cientos de milisegundos en ejecutarse, dependiendo de la tasa de baudios. No se debería invocar mientras está activo uno de los movimientos programados o la secuencia.

El comando SSCAT puede tardar cientos de milisegundos en ejecutarse. No se debería invocar mientras está activo uno de los movimientos programados o la secuencia.

Si un comando SSDEL o SSCAT está en ejecución, no debe apagar el SSC-32 hasta que haya finalizado. Para determinar si el comando se ha completado, envíe un comando y espere la respuesta.

Cada vez que se cambia la cadena de inicio, la ubicación de la memoria EEPROM utilizada para almacenar el valor, experimentará un pequeño gasto de espacio. El número máximo de escrituras es 100.000. No configure su software para que cambie rápidamente la cadena de inicio, ya que se podría dañar la memoria EEPROM en el procesador ATMega168.

Ejemplos de cadena de inicio	
Comando	**Resultado**
SSDEL 255 <cr> SS <cr>	""<cr>
SSCAT #0P2000T5000;<cr> SS <cr>	"#0P2000T5000;"<cr>
SSCAT XXXX<cr> SSC <cr>	"#0P2000T5000;XXXX"<cr>
SSDEL 4 <cr> SS <cr>	"#0P2000T5000;"<cr>
SSDEL 6 <cr> SS <cr>	"#0P2000"<cr>
SSCAT #1P1000T4000;PL0SQ5;<cr> SS <cr>	"#0P2000#1P1000T4000;PL0SQ5;"<cr>

Ejemplos adicionales

Ejemplos adicionales	
Comando	**Resultado**
RDFLT	Establecer todos los registros a los valores predeterminados
SSDEL 255	Borrar la cadena de inicio
R0	Mostrar registro 0
R0=2 R1=2000 R2=1000	Establecer el retardo de transmisión a 2000uS y la velocidad de transmisión a 1000uS. (R0=2: Bit 1 de R0 activa el retardo/velocidad de TX)
R0=12 R32=50 R64=1000	Establece la compensación de pulsos para el servo 0 a 50 y el ancho de pulso inicial a 1000. (Bits 2 y 3 de R0 activan la compensación de pulso y el ancho de pulso.)
R0=13 SSDEL 255 SSCAT #0P1500T5000;	Mover R0 lentamente con un ancho de pulso de 1500 en el inicio. Tenga en cuenta que el ancho de pulso se especifica como en el ejemplo anterior. (Bit 0 de R0 activa la cadena de inicio.)
SS	Mostrar la cadena de inicio actual.

Comprobación del controlador

La forma más fácil de comprobar el controlador es utilizar LynxTerm, nuestro programa de terminal gratuito. Puede descargar LynxTerm aquí.

Una vez instalado el programa, seleccione el puerto COM en la lista desplegable. Esto funciona con los adaptadores del puerto USB a Serie. Instale los jumpers para la tasa de 115,2k baudios y los dos jumpers de activación del puerto serie DB9. Conecte el cable DB9 M/H directo desde el ordenador y el controlador.

Instale dos servos, uno en el canal 0 y otro en el canal 1.

Active la alimentación de SSC-32 (Lógica y servos) y compruebe que se ilumina el indicador verde.

Haga clic en la ventana de terminal y escriba lo siguiente. Recuerde que <cr> equivale a pulsar la tecla Enter.

#0 P1500 #1 P1500 <cr>

Debería comprobar que ambos servos se mantienen en su posición central. El indicador se apaga. Tan sólo se iluminará cuando el controlador esté recibiendo datos. Escriba lo siguiente:

#0 P750 #1 P1000 T3000 <cr>

Debería comprobar que el servo 0 se mueve lentamente hacia la derecha y el servo 1 se mueve hacia la izquierda hacia la izquierda. Alcanzarán sus destinos al mismo tiempo aunque se muevan con ritmos diferentes.

Ahora compruebe la consulta del estado de movimiento. Escriba lo siguiente:

#0 P750 <cr>

Escriba la línea siguiente. Esto hará que el servo tarda 10 segundos en hacer el movimiento completo.

#0 P2250 T10000 <cr>

Mientras el servo esté en movimiento, escriba lo siguiente:

Q <cr>

Cuando el servo esté en movimiento, el controlador devolverá un "+". Devolverá un "." cuando haya alcanzado su destino.

Para experimentar con el argumento de velocidad, pruebe lo siguiente:

#0 P750 S1000 <cr>

Esto moverá el servo desde 2250 hasta 750 (aprox. 170°) en 1,5 segundos.

Distancia de desplazamiento / Valor de velocidad = Tiempo de desplazamiento

(2250uS-750uS) / (1000uS/seg.) = 1,5 seg.

A continuación, pruebe lo siguiente:

#0 P2250 S750 <cr>

Esto moverá el servo desde 750 hasta 2250 (aprox. 170°) en 2,0 segundos.

(2250uS-750uS) / (750uS/seg.) = 2,0 seg.

Los valores de velocidad alrededor de 3500 moverán el servo con la mayor velocidad posible.

Información para resolver problemas

Si los servos se desactivan o dejan de mantener la posición central al mover varios servos de forma simultánea, entonces SSC-32 se ha reseteado. Esto puede verificarse si el indicador verde está encendido fijo después de ordenar que se muevan los servos. El indicador verde no es el indicador encendido, sino un indicador de estado. Cuando se enciende SSC-32, el indicador se iluminará de forma fija. Permanecerá encendido hasta que la unidad reciba un comando serie válido, y se apagará y parpadeará sólo cuando se reciban datos serie.

SSC-32 tiene dos entradas de alimentación. La alimentación lógica (VL) alimenta el microcontrolador y su circuito a través de un regulador de 5v CC. La alimentación de servo (VS) alimenta directamente a los servos. En el modo de alimentación sencilla (predeterminada), los jumpers VS1=VL proporcionará alimentación al regulador CL de 5v CC desde el terminal VS. Funciona mejor con una batería y también con un pack de batería, siempre que la tensión no caiga demasiado. Sin embargo, si cae en exceso, la tensión del microcontrolador se interrumpirá y el SSC-32 se reseteará. Para resolver este problema, quite el jumper VS1=VL y conecte una batería de 9 v CC a la entrada VL. Se aíslan las alimentaciones del servo y lógica para que no se afecten mutuamente.

El modo de alimentación sencilla generalmente es más seguro para las condiciones siguientes:

- VS de 7,2 v CC 2800mAh NiCad o pack de pilas de NiMH para hasta 24 servos.
- VS de 7,2 v CC 2800mAh LiPo o pack de pilas de NiMH para hasta 24 servos.
- VS de 6.0 v CC 1600mAh NiCad p pack de pilas de NiMH para hasta 18 servos.
- VS de alimentador de 6.0 v CC de 2.0 amperios para hasta 8 servos.

Tenga en cuenta que hay directrices generales y algunas excepciones. Lo único que puede causar este efecto es un sistema de alimentación insuficiente. Se puede producir el mismo problema si los cables encargados de trasmitir la corriente son demasiado pequeños, las conexiones tienen cables pelados o doblados o se utilizan soportes de las baterías de poca calidad. El 99% de los problemas del SSC-32 está relacionado con la alimentación. Si detectase movimientos erróneos o inestables de los servos, debería comprobar el sistema de alimentación.

Ejemplos básicos de programación

Ejemplo de un solo bip

```
' Atom / SSC-32 Test
' Configure the SSC-32 for 38.4k baud.

' Note, a | means the line continues onto the next line.
' Note, a ' means the line is a comment, and will not be compiled.

servo0pw var word
movetime var word

servo0pw = 1000
movetime = 2500

start:
servo0pw = 1000
serout p0,i38400,["#0P",DEC servo0pw,"T",DEC movetime,13]
pause 2500
servo0pw = 2000
serout p0,i38400,["#0P",DEC servo0pw,"T",DEC movetime,13]
pause 2500
goto start

' Biped example program.
aa  var byte '<- general purpose variable.
rax var word '<- right ankle side-to-side. On pin0
ray var word '<- right ankle front-to-back. On pin1
rkn var word '<- right knee. On pin2
rhx var word '<- right hip front-to-back. On pin3
rhy var word '<- right hip side-to-side. On pin4
lax var word '<- left ankle side-to-side. On pin5
lay var word '<- left ankle front-to-back. On pin6
lkn var word '<- left knee. On pin7
lhx var word '<- left hip front-to-back. On pin8
lhy var word '<- left hip side-to-side. On pin9
ttm var word '<- time to take for the current move.

' First command to turn the servos on.
for aa=0 to 9
serout p0,i38400,["#", DEC2 aa\1, "P", DEC 1500, 13]
next

start:
' First position for step sequence, and time to move, put in your values here.
rax=1400: ray=1400: rkn=1400: rhx=1400: rhy=1400
lax=1400: lay=1400: lkn=1400: lhx=1400: lhy=1400
ttm=1000
gosub send_data
pause ttm

' Second position for step sequence, and time to move, put in your values here.
rax=1600: ray=1600: rkn=1600: rhx=1600: rhy=1600
lax=1600: lay=1600: lkn=1600: lhx=1600: lhy=1600
ttm=1000
gosub send_data
pause ttm

' Third...

' Forth...

' Etc...

goto start
```

```
' This sends the data to the SSC-32. The serout is all one line. Use |!
send_data:
serout p0,i38400,["#0P",DEC rax,"#1P",DEC ray,"#2P",DEC rkn,"#3P",DEC |
rhx,"#4P",DEC rhy,"#5P",DEC lax,"#6P",DEC lay,"#7P",DEC lkn,"#8P",DEC |
lhx,"#9P",DEC lhy,"T",DEC ttm,13]
return
```